# Surface Mesh Generation for Dirty Geometries by Shrink Wrapping using Cartesian Grid Approach

Y. K. Lee[1], Chin K. Lim[2], Hamid Ghazialam[3], Harsh Vardhan[4] and Erling Eklund[5]

[1] Fluent USA Inc., Evanston, IL 60201 ykl@fluent.com
[2] Fluent USA Inc., Austin, TX 78746, ckl@fluent.com
[3] Fluent USA Inc., Lebanon, NH 03766, hsg@fluent.com
[4] Fluent India Pvt Ltd., Hinjewadi, Pune 411057, India, hxv@fluent.co.in
[5] Fluent France S.A., 78180 Montigny le Bretonneux, France, erling@fluent.fr

**Summary**. A Cartesian shrink wrapping technique has been investigated in this study to construct triangular surface meshes for three-dimensional dirty geometries. The geometries dealt in this paper are defined by faceted representation with dirtiness such as non-conforming edges, gaps and overlaps. The objective of the proposed technique is to deliver a way constructing triangular surface meshes for upstream solutions in design processes without extensive labors for healing dirtiness in complicated dirty geometries. A Cartesian grid is overlaid onto the dirty geometries and its cells are adaptively refined until target resolution is achieved while recording intersections with geometric facets in cells. An initial watertight shell called the wrapper surface is constructed by selectively extracting the boundary sides of intersected cells. The wrapper surface is improved by a subsequence of operations such as projecting nodes onto geometry, adjusting nodes on the geometry and editing local triangular faces to achieve better approximation. The meshes generated using the presented technique may not be geometrically accurate but their quality is good enough to quickly deliver upstream fluid analysis solutions with significantly reduced engineering time for problems of extreme complexity such as the full underhood fluid/thermal analysis for automobiles. Mesh generation experiments have been carried out for complicated geometries and results from some applications are presented in this paper.

## 1. Introduction

Automatic mesh generation has become an essential tool for the finite element or finite volume analyses of practical engineering problems. The geometries of the problems are defined as CAD data and access to information stored in CAD data can be provided either through geometric modeling kernel or exporting the data in a format loadable into the target system [BWS03]. The IGES and STEP are most commonly used protocols for exchanging CAD data from one CAD system to the other system. Even though the STEP normally delivers better translation results than IGES does by providing information with representations and global tolerances, immigration of CAD data from one system to the other often results in dirty geometries containing gaps, holes, overlaps and non-conformal edges which do not exist in the native data. The inconsistency of tolerant modeling methods in two systems is one of major reasons causing dirtiness in the imported geometry. Another source of dirty geometries often ignored is the urgent need of upstream solutions in practice. In such cases, engineers have to carry out simulation even before all the parts in the model have not been designed by leaving small voids or using similar parts in their legacy library.

Imbedding the mesh generation system in CAD systems can be a way to avoid the dirtiness from happening, since the meshing operation can be applied on the native data without translation. The cost for imbedding is extremely expensive and it cannot be done often due to issues other than technical ones. An alternative for imbedding is to develop the mesh generation system inter-operatable with CAD systems. The development cost for this framework is lower than that of imbedding. However, this approach needs both the CAD and the mesh generation system to be available locally. In addition, the CAD data and the mesh data should be managed separately afterward.

Most conventional approach is to heal the dirtiness by directly editing geometric entities of dirty geometries. Manual healing for dirty geometric models is very labor intensive and time consuming as engineers should destructively replace the old geometric entities using sophisticated reasoning. An investigation was carried out for detail suppression using topological modifications and the concept of virtual topology was suggested to define topologies of problems using underlying dirty geometries without extensive editing [SBC97]. Yet cleaning up dirty geometries requires considerable user interactions.

While the previous algorithms intend to fix dirtiness in CAD models, the scope of this study is focused on generation of meshes without altering the input geometries. A major bottleneck from dirty geometries to 3D fluid simulations lies in constructing watertight surface meshes as the subsequent tetrahedral volume mesh generation is relatively straightforward. The Cartesian grid approach and shrink wrapping technique are employed to tackle the problem in the present study. The shrink wrapping approach was proposed by Kobbelt et al. [KVL99] In their approach, a plastic membrane is wrapped around an object and shrunk either by heating the material or by evacuating the air from the space in between the membrane and the object's surface. Theoretically, the plastic skin provides an exact imprint of the given geometry at the end of process. The problems in applying their technique to dirty geometry meshing are, first, the construction of wrapping membrane surface is difficult and, second, the projection operator may not always provide reliable results to compute traction and relaxation due to the nature of dirtiness. Thus, the Cartesian grid approach is introduced to construct the wrapping (or wrapper) surface and geometric modification on the surface is taken to improve closeness of the wrapper surface to the given geometry.

The Cartesian grid generation is a well-established technique. Basically, it overlays an axis-aligned structured grid onto the geometry of a problem and takes a part of the grid, which is in the region of interest. Later, the nodes near the input geometry can be repositioned or local structure of mesh can be modified. The octree technique for constructing tetrahedral meshes can be classified into this technique as it checks intersections between the cells in the tree and the input geometry and refines each cell in the region of interest based on intersecting pattern [She85]. Schneider is an early investigator with full hexahedral elements for 3D volume mesh generation [Sch95]. Aftosmis and his colleague presented a technique clipping geometric facets with Cartesian cells and using such information in fluid simulations [ABM97] and an extended technology has been used in extracting large scale models such as building geometries using shape profiles [WCG05]. Wang extended the application of this technique to dirty geometry cases [ZF02]. He pointed out that the Cartesian mesh generation is tolerable to dirtiness of geometries as long as the region of interest can be classified. And the zigzag boundary of the meshes can be improved using the best approximation available. There have been active development works to exploit such technology in extremely complicated applications with many components and some dirtiness in industrial field [CDA06, CEI06, CFD06] but few articles have been published in the open literature. More recently, Boschoff and Pavic carried out an extensive research for extracting clean surface meshes from architectural models containing penetrating and touching components [BP05].

## 2. Outline of the Proposed Algorithm

The major difficulty in generating 3D volume meshes for dirty geometries for fluid simulations with traditional mesh generation tools is to make the dirty geometry clean so that the meshing tools can be applied step by step, for instance, from edge meshing to surface meshing and, then, to volume meshing. The main idea of the technique presented is (1) to construct a watertight surface mesh for a dirty geometry by extracting a closed surface representation from underlying Cartesian grid and (2) improve the surface mesh while maintaining water-tightness for better approximation to the original geometry. The watertight faceted representation (or surface mesh) may not precisely represent geometric details of the model but, at least, provides a quality surface mesh for the subsequent tetrahedral volume mesh generation and, eventually, the upstream fluid simulation. The two phases can be concisely summarized as follows.

*Overlaying Cartesian grid and extracting wrapping surface*: A simple Cartesian grid of small number of cells is overlaid to the input geometry. The cells in the grid are refined until all of them satisfy criteria. The refinement for Cartesian cells is carried out while checking intersections between cells and the input geometry. Given the intersecting cells, all the non-intersected cells are classified into regions bounded by the intersecting cells. By collecting outer front of intersecting cells, it is possible to construct a watertight surface, so-called the wrapper surface, for each region.

*Modifications of wrapper surface*: In general, the initial wrapper surface represents the topology of a volume collectively defined by the input surfaces. However, the detailed geometry of the wrapper surface is far different from that of the input volume. The wrapper surface is modified by adjusting nodal positions or editing local connectivity of nodes to get better geometric approximation to the input geometry.

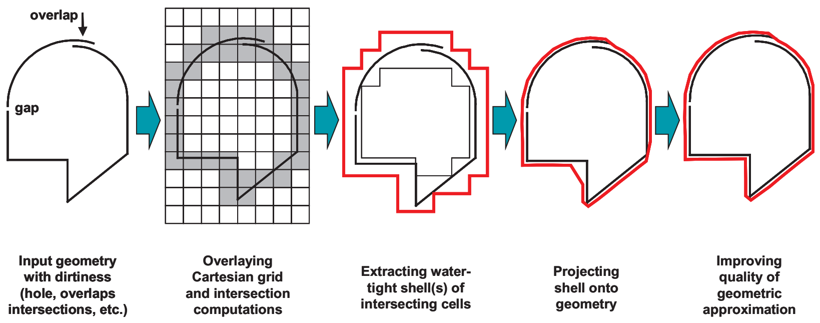

overlap
gap
Input geometry with dirtiness (hole, overlaps intersections, etc.)
Overlaying Cartesian grid and intersection computations
Extracting water-tight shell(s) of intersecting cells
Projecting shell onto geometry
Improving quality of geometric approximation

**Fig. 1.** Schematic description of wrapping procedure

As briefly demonstrated in Figure 1, it is always possible to construct a watertight surface approximating the volume boundary by extracting the connected boundary faces of the intersected cells. The typical case in which the previous statement is not true is when a cell completely falls between gaps, so-called invisible gap [WS02]. However, geometries in practice are reasonably well defined and the failure would not happen if cells around gaps are larger than gap distance.

## 3. Initial Wrapper Surface Generation

The first phase of the current technique is to construct a rough approximation for the input dirty geometry, the wrapper surface. This section describes how to generate the initial wrapper surface using the Cartesian grid technique.

### 3.1 Initial Cartesian Grid Generation

The faceted representation such as the stereolithography (STL) format is used to define the input geometry in this study. The STL format is popular due to its simplicity and portability. The facets in the geometry are stored in a search tree to be used for the intersection checks and projection in this study.

A uniform Cartesian grid is overlaid on the input geometry and its cells are adaptively refined later. The Cartesian grid is defined with cells and faces. In this paper, cells are defined with their six faces, locations and dimensions. Each face records its left and right neighbor cells. They also store references to child faces in case of refinement. Vertices are not used because they are not necessary [ABM97].

### 3.2 Adaptation of the Grid

Starting from the initial grid, an adaptive grid is constructed by gradually refining cells until all cells satisfy give size criteria. The size functions [ZBS02] are used to define the desired local sizes. While the curvature size function generally regards quality of geometric approximation, the proximity size functions may result topological difference of surface meshes obtained. For example as shown in Figure 2, insufficient refinement between gaps in a single connected simple geometry results a double connected representation. The problem is resolved only after two more refinement for cells (see Figure 2(b)). In 2D cases, cells of size $2G/\sqrt{2}$ are required to ensure that the gaps are resolved regardless to orientations and translations. Similarly, the factor becomes $2G/\sqrt{3}$ in 3D cases. The maximum difference of refinement between neighboring cells is limited to 1 both for simplicity in the data structure and smooth transition of cell sizes.

## 3.3 Extracting Initial Wrapper Surface

In the adapted Cartesian grid, there may be several groups of contiguous non-intersecting cells, the *regions*. A region may represent a virtual volume. In complicated models, there are multiple virtual volumes and geometric complexity often results exceptional cases.

Let us paint cells intersecting the input geometry in a simple grid shown in Figure 3(a). Naturally, those cells form virtual walls which separate non-intersecting cells into regions. Starting from a side of an intersecting cell on the exterior region, a closed shell can be extracted by following neighboring boundary sides of intersecting cells (see Figure 3(b)). There might be exceptional cases as shown in the interior regions causing non-manifold connection and fictitious regions. The intersection status of some cells should be manipulated to resolve such cases.

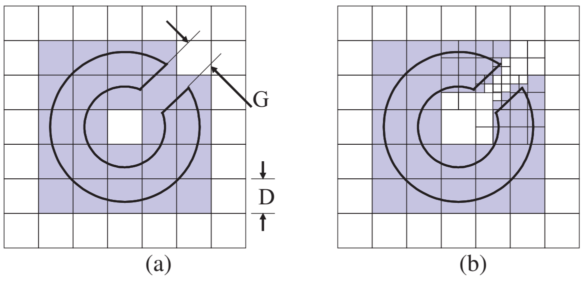


**Fig. 2.** Proximity size function to resolve gaps

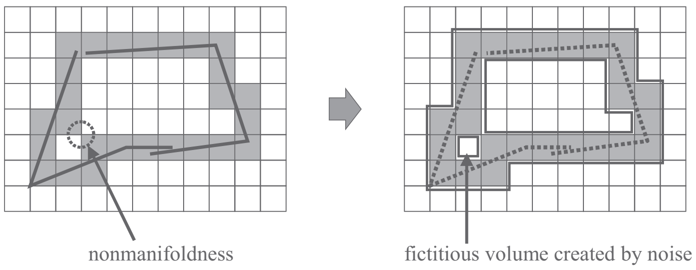


**Fig. 3.** Nonmanifoldness and fictitious region

## 4. Improvement of Wrapper Surfaces

In general, the initial wrapper surface extracted from the Cartesian grid topologically represents the virtual volume that is desired to recover. However, its geometric details, the zigzag configurations, are far different from the desired geometry. The remaining procedure of the developed technique is to improve such poor geometric approximation that it represents the input geometry better. This is done by adjusting node positions and locally editing triangular elements.

### 4.1 Projection of Nodes

The first step for improving the wrapping surface is to move nodes of the zigzagged wrapper surface onto the input geometry. The simplest way is to project the nodes onto the nearest points on the input geometry. However, the nearest point projection may not give desired results as schematically shown in Figure 4.

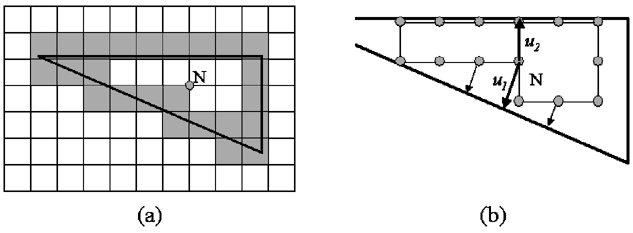


**Fig. 4.** Abnormal projection of node

Due to the nature of the current approach, it is obvious that all the nodes are all inside of the volume. (Or all nodes should be outside of the volume if the exterior wrapper surface was taken.) Thus all nodes should be projected to either outward or inward. An approximated normal vector, $\tilde{\mathbf{n}}_v^N$, at a node, $N$, is computed as

$$\mathbf{n}_v^N = \frac{\sum_i \mathbf{n}_i^F}{\left\| \sum_i \mathbf{n}_i^F \right\|}, \quad \tilde{\mathbf{n}}_v^N = \frac{\mathbf{n}_v^N + \sum_j \mathbf{n}_j^N}{\left\| \mathbf{n}_v^N + \sum_j \mathbf{n}_j^N \right\|} \tag{1}$$

where $\mathbf{n}_i^F$ is a normal vector of its adjacent face and $\mathbf{n}_v^N$ is the averaged normal vector of them. When a node is found to have the sign of inner product of its approximated normal vector and projection vector that is inconsistent to those of its neighboring nodes, some preconditioning such as a weighted Laplacian smoothing should be done to prevent invalid projection.

In many practical cases, there are distinctive features on the boundary such as sharp edges and boundary between two zones. Often, it is highly desired to restore such features in the wrapper surface. In addition to the projection onto the surface, nodes in the vicinity of feature curves in the geometry are projected onto the feature curves. For a given feature curve, the closest nodes from the two ends are found on the wrapper surface and a path between the nodes is traced by using the feature curve as a guide. This procedure is basically traveling through mesh edges from one end node to the other while comparing distance to the feature curve and dot product of the edge vectors and the feature curve tangential vector. The tracing may fail to identify a reliable path and further investigation is undergoing. Even after identifying a reliable path, it is often observed that the compulsive projection of the nodes onto the feature curves deteriorates the configurations of neighboring faces. The projection of nodes is carried out in an incremental manner while checking validity of neighboring faces.

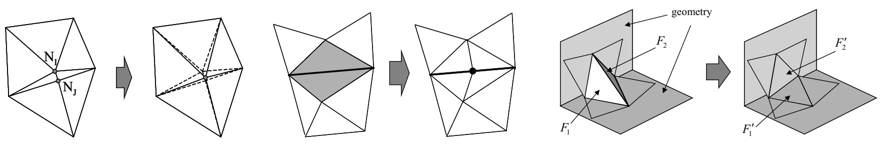
## 4.2 Editing Local Connectivities

In general, a finer initial wrapper surface provides better approximation for the input geometry after the projection. However, there may be very slender and skewed triangles. A set of standard local modification operations for triangles is used to improve such undesirable configurations further.

*Edge collapsing*: An edge that is unacceptably shorter than its neighboring edges is removed by merging its two end nodes as shown in Figure 5 (a). Also, this operation is used to coarsen surface meshes [GH97].

*Edge splitting*: If an edge is too long comparing to its neighboring edges, then a new node is introduced at its center and its two adjacent faces are divided into four triangles in Figure 5(b). The edge splitting is also used to improve skewed triangles.

*Edge swapping*: The edge swapping shown in Figure 5(c) can be very effective to improve the closeness of the faces to the geometry as well as to improve skewness of slender triangles.



(a) Edge collapsing (b) Edge splitting (c) Edge swapping

**Fig. 5.** Local mesh editing

### 4.3 Node Smoothing

Mostly, a weighted Laplacian smoothing [Her76] is used to improve skewed triangles by repositioning nodes. The nodes are projected back to the input geometry afterward. When a large number of iterations were applied to improve triangular quality, excessive smoothing deteriorates details of configurations and makes projection of nodes back to the geometry harder by moving nodes too far from the geometry. In such cases, enhanced smoothing techniques preserving features and volumes can be very effective [Tau95, VMM99, ZF02].

### 4.4 Zone Partitioning

The wrapped surface can be separated into several regions based on the underlying surfaces. The closest input surfaces are checked for every faces and faces of same corresponding input surface are grouped and separated into a zone. This is particularly useful when boundary conditions should be applied for certain zones in the fluid simulations later. In many cases, there are many small fictitious zones after initial separation due to the fact that the wrapper zone boundaries do not exactly follow those of input surfaces. In such cases, the faces in small islands are redistributed to neighboring larger zones.

## 5. Mesh Generation Examples

The presented algorithms are implemented in TGrid, a preprocessor for FLUENT solver. The developed mesh generator has been exercised on a set of example problems and the results are presented in the following sections. The presented examples include a simple geometry to graphically demonstrate the meshing procedure as well as fairly complicated geometries having intersecting facets to validate effectiveness of the proposed technique.

## 5.1 Transport Aircraft: General Wrapping Procedure

A relatively simple example is taken in this section to demonstrate the typical mesh generation procedure developed.

The model is an artificial transportation aircraft which is composed of five separate parts – fuselage, two wings and two sheet metal parts joining the fuselage and wings. An initial Cartesian grid is overlaid to the input geometry shown in Figure 6(a) and the configurations of intersected cells can be shown as in Figure 6(b). The connected exterior faces of the cells were extracted and the faces were triangulated based on predefined pattern in terms of hanging node configurations. Figure 6(c) shows the initial shrink wrapped surface after projection. The configuration in Figure 6(c) was improved using the smoothing, local mesh editing, etc. The final mesh is presented in Figure 6(d) after the zone partitioning.

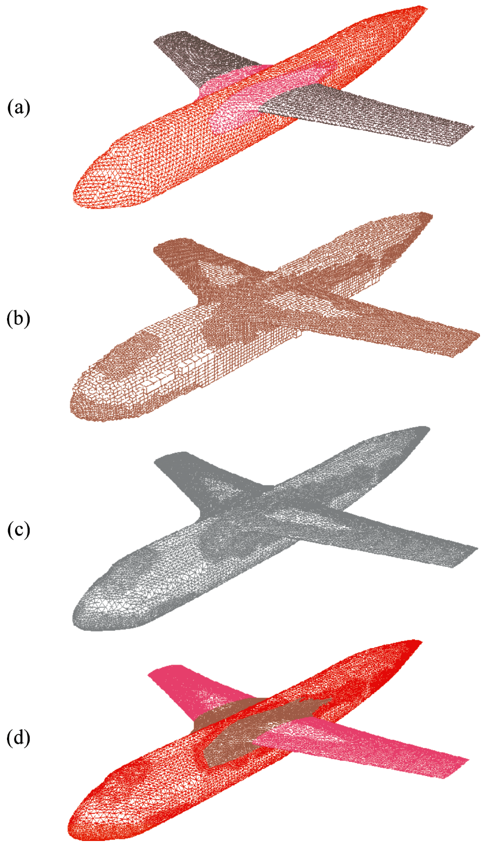


**Fig. 6.** Typical mesh generation procedure. (a) Input geometry; (b) Intersected Cartesian cells; (c) Initial wrapper surface; (d) Final wrapper surface after improvements

The following figure shows the minor dirtiness of the model. The part joining the wing and the fuselage does not share identical nodes and edges with either parts and some portion is penetrating into the fuselage.

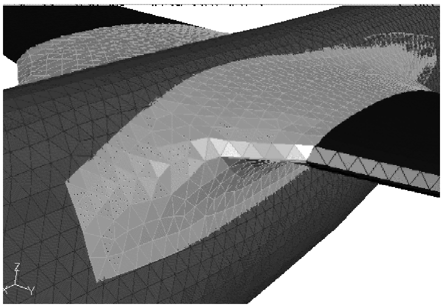
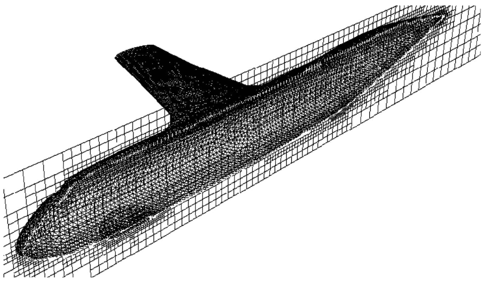
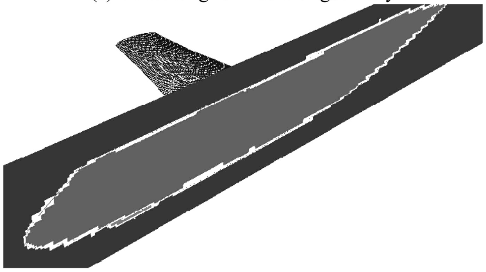


**Fig. 7.** Dirtiness in aircraft model



(a) Cartesian grid lines with geometry



(b) Highlighted intersected cells and inside and outside regions

**Fig. 8.** Cartesian grid lines on cutting plane

Figure 8(a) displays the gridlines of the underlying Cartesian grid on a cutting plane. The white lines are on cells intersecting the input geometric facets. The regions of different shades in Figure 8(b) represent that they are in separated regions bounded by the intersected cells shown as the white region.

In the existence of distinctive edges, so-called feature lines, in the initial geometry, such features can be restored in the final mesh by projecting nodes onto the feature lines as shown in Figure 9.

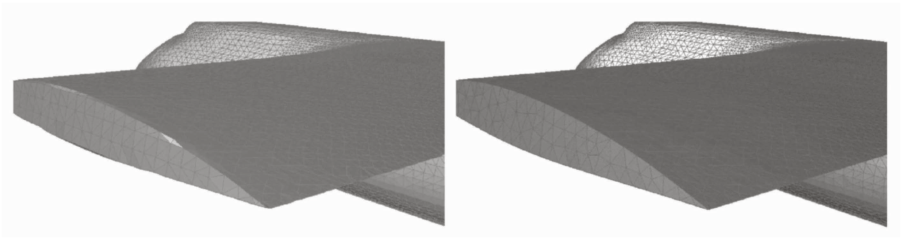
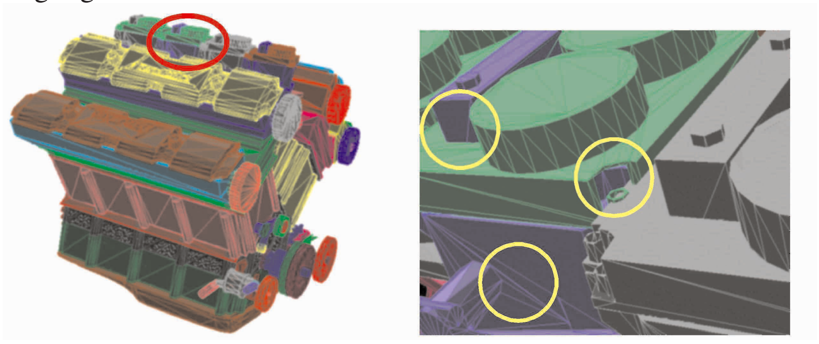


**Fig. 9.** Feature line recovery

## 5.2 Combined Engine Block

The second example in this paper is a V8 engine (see Figure 8(a)). The model contains three types of typical dirtiness to be dealt in dirty geometry meshing, intersections by penetrating parts, holes due to missing parts and unreliable facets due to poor triangulation common in STL files. Figure 8(b) highlights such dirtiness in circles.



(a) Combined engine block. (b) Three types of dirtiness

**Fig. 10.** V8 engine

In general, any holes whose sizes are larger than cells in the Cartesian grid may result leakages in wrapping procedure. Thus such holes are desired to be filled prior to the wrapping operation. However, it is almost impossible to identify such holes in many complicated models in practice. If a model is wrapped with leakages due to such holes, the resulting wrapped surface is folded at a certain location, where the leakage occurs and the wrapper surface propagates into the inside of the volume. In such case, a pair of faces can be chosen and the path between them can be tracked as shown in Figure 11 and the location of the hole should lie on the path. The hole should be resolved by adding additional facets manually.

Figure 12 shows a contour plot displaying the distance of each face center to the underlying input geometry. This plot can be exploited to quickly identify locations where higher resolutions with smaller cells are required if necessary.

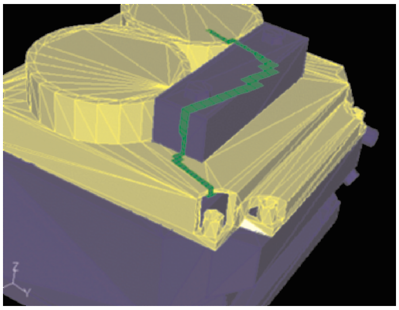


**Fig. 11.** Hole identification with path tracking

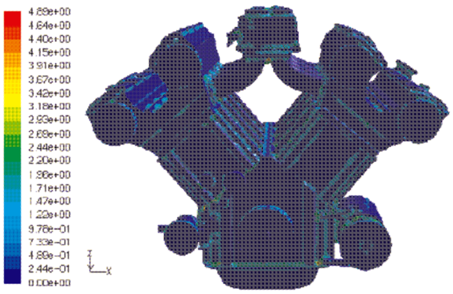


**Fig. 12.** Distribution of distance between face centers and underlying geometry

Figure 13(a) illustrates noisy zigzag configurations along the zone boundaries after partitioning mentioned in the section 4.6. The difficult cases are to recovery smooth boundary when there is no explicit boundary between the two zones in the underlying geometry, for instance, a pair of parts penetrating each other. In such cases, the nodes on the zigzag boundary can be projected onto both parts alternatively and the boundary forms a smooth line as shown in Figure 13(b).

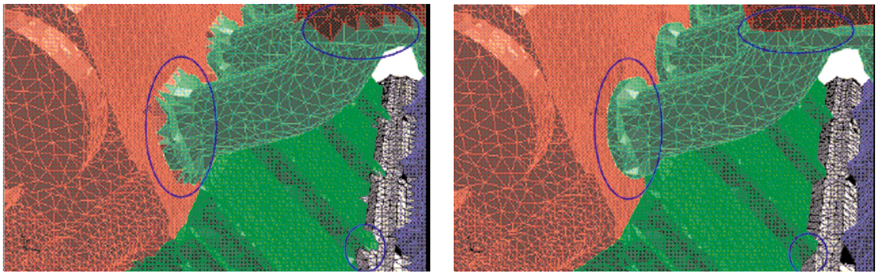
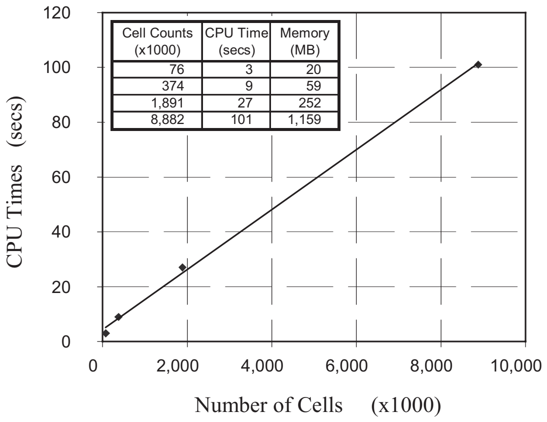


**Fig. 13.** Zone partitioning and boundary recovery

The CPU time complexity for Cartesian grid generation and region separation is shown in Figure 14 with a linear trend line. The experiment was done using a Linux machine with 2 Pentium 4 (3.4GHz) processors. The CPU times spent for subsequent modifications such as smoothing, swapping and coarsening were excluded as they are interactive operations which are triggered one after the other. From the latest refinement stage of 8,882k cells, 4.12 million triangles were extracted into the initial wrapper surface and it took 298 seconds to extract those triangles, incrementally project onto the input geometry and improve severely skewed triangles by edge collapsing and swapping.



| Cell Counts (x1000) | CPU Time (secs) | Memory (MB) |
|---|---|---|
| 76 | 3 | 20 |
| 374 | 9 | 59 |
| 1,891 | 27 | 252 |
| 8,882 | 101 | 1,159 |

**Fig. 14.** CPU time complexity for Cartesian grid generation and region generation

## 5.3 Meshing For Underhood and External Aero-Simulation

The example in this section is taken to illustrate an application of the developed mesh generator for a model of industrial complexity and to discuss the turnaround time in industrial applications. The model for a whole truck body is presented in Figure 15(a) and it contains more than 1250 assembly parts including ones in underhood shown in Figure 15(b). The original geometry contains 473156 facets with 237761 vertices.

Several resolution levels have been tested and numerical experiment carried out using a AMD 64bit Opteron machine shows that the developed mesh generator takes 3.5 GB for 10 million cells and spends 1 hour to generate 60 million cells with intersection checks. The truck body was placed in an artificial wind tunnel and wrapped with it. The final wrapper surface consists of roughly 2 million faces and 1 million nodes after coarsening. It

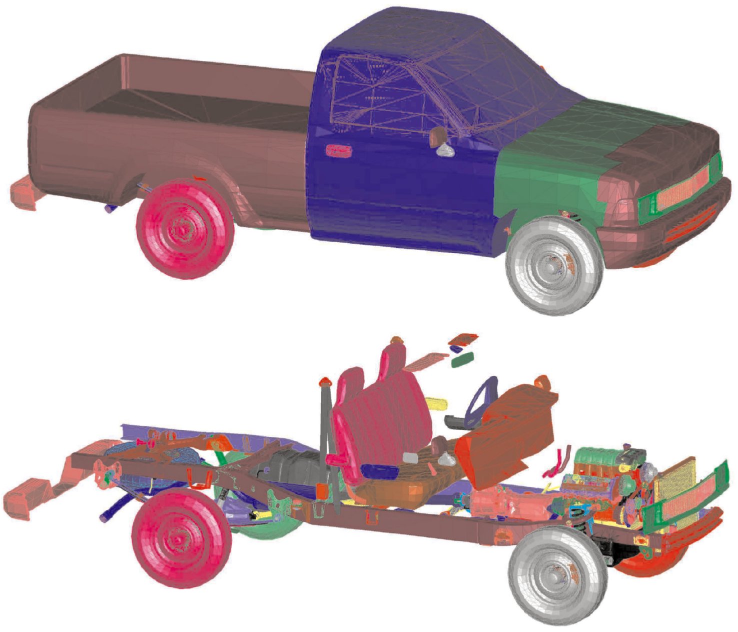


**Fig. 15**. Truck model for underhood thermal management simulation

took roughly 1 man week to construct the wrapped surface mesh ready for tetrahedral volumetric mesh generation from the initial STL files including hole filling and tailored mesh generation and connection for critical parts such as the heat exchanger.

The volume mesh generation was carried out mainly with tetrahedral elements and wedge prism layers were applied to some parts. Several failures occurred due to poor triangular quality and excessively close proximity between opposing faces. The final volume mesh used for the fluid simulation contains 6.8 million cells with 0.7 million interior nodes with the boundary entities mentioned prior. Figure 17 shows streamlines and pressure distributions obtained by the simulation.

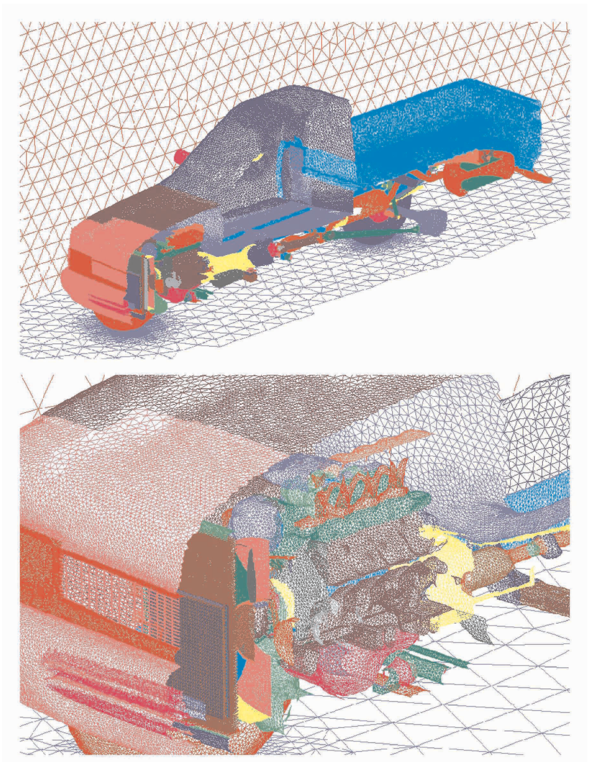


**Fig. 16.** Configurations of final wrapper surface for 3D volume mesh generation

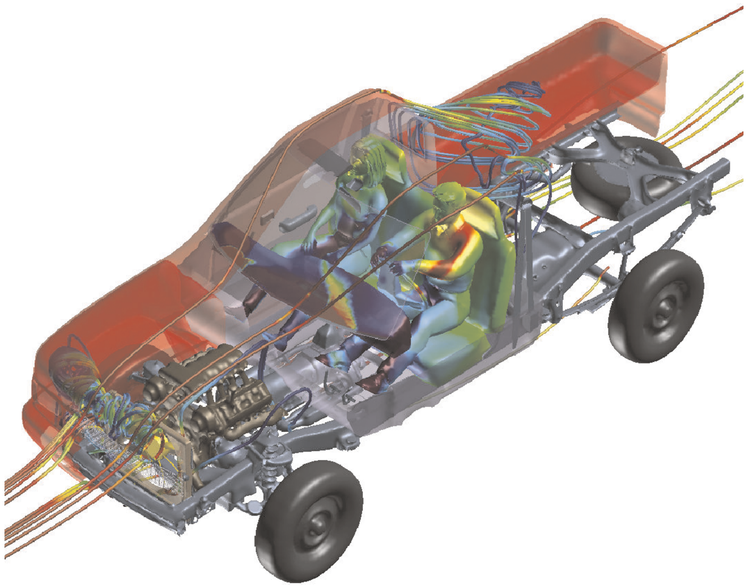


**Fig. 17.** Streamlines and pressure distributions from CFD simulations

## 6. Discussions and Future Works

A mesh generation technique has been investigated to construct triangular 3D surface meshes for dirty geometries and applications of the developed mesh generator have been presented to demonstrate effectiveness of the proposed technique. The proposed algorithm constructs initial watertight triangular meshes called the wrapper surfaces from the Cartesian grid overlaid onto the input dirty geometries by analyzing intersections of its cells. The final wrapper surface is obtained by gradually improving until a quality surface mesh is achieved to proceed to the subsequent tetrahedral volumetric mesh generation with. As the intersection check in Cartesian grid generation is tolerant to geometric dirtiness to a certain extent, it is possible to reduce a significant amount of user interactions that were necessary in traditional geometric healing.

Further works are under investigation in some areas. For example, it will be necessary to improve memory efficiency to exploit the presented technique on extreme applications such as combined fluid simulations of external aero, underhood thermal management and cabin HVAC. The cells in the presented study are represented a face-based unstructured data structure. A comparative study is undergoing using the octree data structure. Also, several key improvements should be made to deliver special need to construct meshes for volumes with thin surfaces attached to. Automated hole fixing is another challenging issue. The two major problems are, first, to identify a hole and, second, resolve it. According to our experience, the hole can be an artifact one caused by missing faces as well as structural one, for example, a hole on a sheet metal component. The former is relatively easier to detect due to the existence of free mesh edges each having only one connected face. The later often appears clean as the thin sheet metal is defined very close two surfaces that are fully connected to neighboring surfaces. In the existence of holes, the traditional way is to patch them with extra triangles manually. An interesting alternative is to coarsen the cells around a hole so that the hole does not appear in the wrapper surface. Further study will be carried out for this issue.

## References

[ABM97] M. J. Aftosmis, M. J. Berger and J. E. Melton (1997) Robust and efficient Cartesian mesh generation for component-based geometry, AIAA Paper 97-0196

[BPK05] S. Bischoff, D. Pavic and L. Kobbelt (2005) Automatic restoration of polygonal models, ACM Trans. Graphics, 24(4), pp. 1332-1352.

[BWS03] M. W. Beall, J. Walsh and Mark S. Shephard (2003) Accessing CAD geometry for mesh generation. Proc. 12th International Meshing Roundtable, pp. 33-42

[CDA06] CD-Adapco (2006) PROSTAR Automatic Meshing, http://www.adapco.com/SoftwareProducts/ proam-htm.htm

[CEI06] CEI Ensight (2006) Harpoon: The Extreme Mesher, http://www.ensight.com/products/harpoon.html

[CFD06] CFDRC, CFD-VisCART, http://www.cfdrc.com/serv_prod/cfd_multiphysics/software/ace/viscart.html

[GH97] M. Garland and P. S. Heckbert (1997) Surface simplification using quadric error metrics, Proc. 24th Annual Conf. on Computer Graphics and Interactive Techniques, pp. 209-216

[Her76] L. R. Hermann (1976) Laplacian-Isoparametric Grid Generation Scheme, J. of the Engineering Mechanics Division of the American Society of Civil Engineers, 102, pp. 749-756

[KVL99] L. Kobbelt, J. Vorsatz, U. Labsik and H.P. Seidel (1999) A shring wrapping approach to remeshing polygonal surfaces, Computer Graphics Forum, 18(3), pp. 119-130

[PER04] M. Peric (2004) Simulation of flows in complex geometries: New meshing and Solution Methods, Proc. NAFEMS Seminar: "Simulation of Complex Flows (CFD) - Application and Trends", Niedernhausen/Wiesbaden, Germany

[She85] M. S. Shephard (1985) Automatic and adaptive mesh generation, IEEE Trans. Magnetics, 21, pp. 2482-2489

[SBC97] A. Sheffer, T. Blacker, J. Clements and M. Bercovier (1997) Virtual Topology Operators for Meshing, Proceedings, 6th International Meshing Roundtable, pp. 49-66

[Sch95] R. Schneiders (1995) Automatic Generation of Hexahedral Finite Element Meshes, Proceedings, 4th Int. Meshing Roundtable, pp. 103-114

[SWC00] J. P. Steinbrenner, N. J. Wyman and J. R. Chawner (2000) Fast Surface Meshing on Imperfect CAD Models, Proc. 9th Int. Meshing Roundtable, pp. 33-41

[Tau95] G. Taubin (1995) Curve and Surface Smoothing without Shrinkage, Proc. 5th Int. Conf. Computer Vision, pp. 852-857

[VMM99] J. Vollmer, R. Mencl and H. Müller (1999) Improved Laplacian smoothing of noisy surface meshes, Computer Graphics Forum, 18(3), pp. 131-138.

[WCG05] A. Wissink, K. Chand, B. Gunney, C. Kapfer, M. Berger, B. Kosovic, S. Chan and F. Chow (2005) Adaptive urban dispersion integrated model, 8th American Meteorological Society Annual Meeting, Atlanta, GA.

[WS02] J. Z. Wang and K. Srinivasan (2002) An adaptive Cartesian grid generation method for 'Dirty' geometry, Int. J. Numer. Meth. Fluids, 39, pp. 703-717

[ZBS02] J. Zhu, T. Blacker and R. Smith (2002) Background Overlay Grid Size Functions, Proc. 11th Int. Meshing Roundtable, pp. 65-74

[ZF02] H. Zhang, E. Fiume (2002) Mesh smoothing with shape or feature preservation, Proc. Computer Graphics International 2002, pp. 167-182.